openpgp
Internet-Draft
Intended status: Informational
Expires: February 23, 2020

D. Gillmor
ACLU
August 22, 2019

Abuse-Resistant OpenPGP Keystores
draft-dkg-openpgp-abuse-resistant-keystore-04

Abstract

OpenPGP transferable public keys are composite certificates, made up of primary keys, direct key signatures, user IDs, identity certifications ("signature packets"), subkeys, and so on. They are often assembled by merging multiple certificates that all share the same primary key, and are distributed in public keystores.

Unfortunately, since many keystores permit any third-party to add a certification with any content to any OpenPGP certificate, the assembled/merged form of a certificate can become unwieldy or undistributable. Furthermore, keystores that are searched by user ID or fingerprint can be made unusable for specific searches by public submission of bogus certificates. And finally, keystores open to public submission can also face simple resource exhaustion from flooding with bogus submissions, or legal or other risks from uploads of toxic data.

This draft documents techniques that an archive of OpenPGP certificates can use to mitigate the impact of these various attacks, and the implications of these concerns and mitigations for the rest of the OpenPGP ecosystem.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

- "OpenPGP certificate" (or just "certificate") is used interchangeably with [RFC4880]'s "Transferable Public Key". The term "certificate" refers unambiguously to the entire composite object, unlike "key", which might also be used to refer to a primary key or subkey.

- An "identity certification" (or just "certification") is an [RFC4880] signature packet that covers OpenPGP identity information - that is, any signature packet of type 0x10, 0x11, 0x12, or 0x13. Certifications are said to (try to) "bind" a primary key to a User ID.

- The primary key that makes the certification is known as the "issuer". The primary key over which the certification is made is known as the "subject".

- A "first-party certification" is issued by the primary key of a certificate, and binds itself to a user ID in the certificate. That is, the issuer is the same as the subject. This is sometimes referred to as a "self-sig".

- A "third-party certification" is a made over a primary key and user ID by some other certification-capable primary key. That is, the issuer is different than the subject. The elusive "second-party" is presumed to be the verifier who is trying to interpret the certificate.

- All subkeys are bound to the primary key with an [RFC4880] Subkey Binding Signature. Some subkeys also reciprocate by binding themselves back to the primary key with an [RFC4880] Primary Key Binding Signature. The Primary Key Binding Signature is also known as a "cross-signature" or "cross-sig".

- A "keystore" is any collection of OpenPGP certificates. Keystores typically receive mergeable updates over the course of their lifetime which might add to the set of OpenPGP certificates they hold, or update the certificates.

- "Certificate validation" is the process whereby a user decides whether a given user ID in an OpenPGP certificate is acceptable for use. For example, if the certificate has a user ID of "Alice <alice@example.org>" and the user wants to send an e-mail to "alice@example.org", the mail user agent might want to ensure that

the certificate is valid for this e-mail address before encrypting to it. Some clients may rely on specific keystores for certificate validation, but some keystores (e.g., [SKS]) make no assertions whatsoever about certificate validity, and others offer only very subtle guarantees. See Section 3.4 for more details.

- "Certificate lookup" refers to the retrieval of a set of certificates from a keystore based on the user ID or some substring match of the user ID. See Section 3.3 for more details.

- "Certificate refresh" refers to retrieval of a certificate from a keystore based on the fingerprint of the primary key. See Section 3.1 for more details.

- "Certificate discovery" refers to the retrieval of a set of certificates from a keystore based on the fingerprint or key ID of any key in the certificate. See Section 3.2 for more details.

- A "keyserver" is a particular kind of keystore, typically a means of publicly distributing OpenPGP certificates or updates to them. Examples of keyserver software include [SKS] and [MAILVELOPE-KEYSERVER]. One common HTTP interface for keyservers is [I-D.shaw-openpgp-hkp].

- A "synchronizing keyserver" is a keyserver which gossips with other peers, and typically acts as an append-only log. Such a keyserver is typically useful for certificate lookup, certificate discovery, and certificate refresh (including revocation information). They are typically _not_ useful for certificate validation, since they make no assertions about whether the identities in the certificates they server are accurate. As of the writing of this document, [SKS] is the canonical synchronizing keyserver implementation, though other implementations exist.

- An "e-mail validating keyserver" is a keyserver which attempts to verify the identity in an OpenPGP certificate's user ID by confirming access to the e-mail account, and optionally by confirming access to the secret key. Some implementations permit removal of a certificate by anyone who can prove access to the e-mail address in question. They are useful for certificate lookup based on e-mail address and certificate validation (by users who trust the operator), but some may not be useful for certificate refresh or certificate discovery, since a certificate could be simply replaced by an adversary who also has access to the e-mail address in question. [MAILVELOPE-KEYSERVER] is an example of such a keyserver.

- A "sovereignty-respecting" keystore is one that only distributes data associated with a given certificate that has been explicitly approved by the primary key of that certificate. See Section 8 for more details and example strategies.

- "Cryptographic validity" refers to mathematical evidence that a signature came from the secret key associated with the public key it claims to come from. Note that a certification may be cryptographically valid without the signed data being true (for example, a given certificate with the user ID "Alice <alice@example.org>" might not belong to the person who controls the e-mail address "alice@example.org" even though the self-sig is cryptographically valid). In particular, cryptographic validity for user ID in a certificate is typically insufficient evidence for certificate validation. Also note that knowledge of the public key of the issuer is necessary to determine whether any given signature is cryptographically valid. Some keyservers perform cryptographic validation in some contexts. Other keyservers (like [SKS]) perform no cryptographic validation whatsoever.

- OpenPGP revocations can have "Reason for Revocation" (see [RFC4880]), which can be either "soft" or "hard". The set of "soft" reasons is: "Key is superseded" and "Key is retired and no longer used". All other reasons (and revocations that do not state a reason) are "hard" revocations. See Section 12.1 for more detail.

2. Problem Statement

OpenPGP keystores that handle submissions from the public are subject to a range of attacks by malicious submitters.

This section describes five distinct attacks that public keystores should consider.

2.1. Certificate Flooding

Many public keystores (including both the [SKS] keyserver network and [MAILVELOPE-KEYSERVER]) allow anyone to attach arbitrary data (in the form of third-party certifications) to any certificate, bloating that certificate to the point of being impossible to effectively retrieve. For example, some OpenPGP implementations simply refuse to process certificates larger than a certain size.

This kind of Denial-of-Service attack makes it possible to make someone else's certificate unretrievable from the keystore, preventing certificate lookup, discovery, or refresh. In the case of

a revoked certificate that has been flooded, this potentially leaves the client of the keystore with the compromised certificate in an unrevoked state locally because it was unable to fetch the revocation information.

Additionally, even without malice, OpenPGP certificates can potentially grow without bound.

2.2. User ID Flooding

Public keystores that are used for certificate lookup may also be vulnerable to attacks that flood the space of known user IDs. In particular, if the keystore accepts arbitrary certificates from the public and does no verification of the user IDs, then any client searching for a given user ID may need to review and process an effectively unbounded set of maliciously-submitted certificates to find the non-malicious certificates they are looking for.

For example, if an attacker knows that a given system consults a keystore looking for certificates which match the e-mail address "alice@example.org", the attacker may upload thousands of certificates containing user IDs that match that address. Even if those certificates would not be accepted by a client (e.g., because they were not certified by a known-good authority), the client still has to iterate through all of them in order to find the non-malicious certificates.

User ID flooding is only effective if the keystore offers a lookup interface at all.

2.3. Fingerprint Flooding

A malicious actor who wants to render a certificate unavailable for refresh may generate an arbitrary number of OpenPGP certificates with the targeted primary key attached as a subkey. If they can convince a keystore to accept all of those certificates, and the keystore returns them by subkey match during certificate refresh, then the certificate refresh client will need to spend an arbitrary amount of bandwidth and processing power filtering out the irrelevant data, and may potentially give up before discovering the certificate of interest.

A malicious actor may also want to confuse a certificate discovery request that was targeted at a particular subkey, by binding that subkey to multiple bogus certificates. If these bogus certificates are ingested and redistributed by the keystore, then a certificate discovery client may receive a set of certificates that cannot be adequately distinguished.

2.4. Keystore Flooding

A public keystore that accepts arbitrary OpenPGP material and is append-only is at risk of being overwhelmed by sheer quantity of malicious uploaded packets. This is a risk even if the user ID space is not being deliberately flooded, and if individual certificates are protected from flooding by any of the mechanisms described later in this document.

The keystore itself can become difficult to operate if the total quantity of data is too large, and if it is a synchronizing keyserver, then the quantities of data may impose unsustainable bandwidth costs on the operator as well.

Effectively mitigating against keystore flooding requires either abandoning the append-only property that some keystores prefer, or imposing very strict controls on initial ingestion.

2.5. Toxic Data

Like any large public dataset, it's possible that a keystore ends up hosting some content that is legally actionable in some jurisdictions, including libel, child pornography, material under copyright or other "intellectual property" controls, blasphemy, hate speech, etc.

A public keystore that accepts and redistributes arbitrary content may face risk due to uploads of toxic data.

3. Keystore Interfaces

Some keystores have simple interfaces, like files present in a local filesystem. But many keystores offer an API for certificate retrieval of different types. This section documents a set of useful interactions that a client may have with such a keystore.

They are represented in abstract form, and are not intended to be the full set of interfaces offered by any keystore, but rather a convenient way to think about the operations that make the keystore useful for its clients.

Not all keystores may offer all of these interfaces, or they may offer them in subtly different forms, but clients will nevertheless try to perform something like these operations with keystores that they interact with.

3.1. Certificate Refresh

This is the simplest keystore operation. The client sends the keystore the full fingerprint of the certificate's primary key, and the keystore sends the client the corresponding certificate (or nothing, if the keystore does not contain a certificate with a matching primary key).

```
keystore.cert_refresh(primary_fpr) -> certificate?
```

A client uses certificate refresh to retrieve the full details of a certificate that it already knows about. For example, it might be interested in refreshs to the certificate known to the keystore, including revocations, expiration refreshs, new third-party certifications, etc.

Upon successful refresh, the client SHOULD merge the retrieved certificate with its local copy.

Not all keystores offer this operation. For example, clients cannot use WKD ([I-D.koch-openpgp-webkey-service]) or OPENPGPKEY ([RFC7929]) for certificate refresh.

3.2. Certificate Discovery

If a client is aware of an OpenPGP signature or certification that it cannot verify because it does not know the issuing certificate, it may consult a keystore to try to discover the certificate based on the Issuer or Issuer Fingerprint subpacket in the signature or certification it is trying to validate.

```
keystore.cert_discovery(keyid|fpr) -> certificate_list
```

This is subtly different from certificate refresh (Section 3.1) in three ways:

- it may return more than one certificate (e.g., when multiple certificates share a subkey, or when a primary key on one certificate is a subkey on another)

- it is willing to accept searches by short key ID, not just fingerprint

- it is willing to match against a subkey, not just a primary key

While a certificate discovery client does not initially know the certificate it is looking for, it's possible that the returned

certificate is one that the client already knows about. For example, a new subkey may have been added to a certificate.

Upon successful discovery, the client SHOULD merge any retrieved certificates with discovered local copies (as determined by primary key), and then evaluate the original signature against any retrieved certificate that appears to be valid and reasonable for use in the signing context.

It is unclear what a client should do if multiple certificates do appear to be valid for a given signature, because of ambiguity this represents about the identity of the signer. However, this ambiguity is similar to the ambiguity of a certificate with multiple valid user IDs, which the client already needs to deal with.

Not all keystores offer this operation. For example, clients cannot use WKD ([I-D.koch-openpgp-webkey-service]) or OPENPGPKEY ([RFC7929]) for certificate discovery.

3.3. Certificate Lookup

If a client wants to encrypt a message to a particular e-mail address, or wants to encrypt a backup to some identity that it knows of but does not have a certificate for, it may consult a keystore to discover certificates that claim that identity in their user ID packets. Both [I-D.koch-openpgp-webkey-service] and [I-D.shaw-openpgp-hkp] offer certificate lookup mechanisms.

[RFC4880] User IDs are constrained only in that they are a UTF-8 string, but some conventions govern their practical use. See Section 12.2 for more discussion of some common conventions around user ID structure.

Note that lookup does not necessarily imply user ID or certificate validation. It is entirely possible for a keystore to return a certificate during lookup that the client cannot validate.

Abuse-resistant keystores that offer a lookup interface SHOULD distinguish interfaces that perform full-string-match lookup from interfaces that perform e-mail address based lookup.

3.3.1. Full User ID Lookup

The most straightforward form of certificate lookup asks for the set of all certificates that contain a user ID that exactly and completely matches the query parameter supplied by the client.

```
keystore.cert_lookup(uid) -> certificate_list
```

In its simplest form, this match is done by a simple bytestring comparison. More sophisticated keystores MAY perform the comparison after applying [UNICODE-NORMALIZATION] form NFC to both the "uid" query and the user IDs from the stored certificates.

3.3.2. E-mail Address Lookup

However, some common use cases look for specific patterns in the user ID rather than the entire user ID. Most useful to many existing OpenPGP clients is a lookup by e-mail address.

```
keystore.cert_lookup(addr) -> certificate_list
```

For certificates with a user ID that matches the structure of an [RFC5322] "name-addr" or "addr-spec", a keystore SHOULD extract the "addr-spec" from the user ID, canonicalize it (see Section 12.3), and compare it to the canonicalized form of of the "addr" query parameter.

3.3.3. Other Lookup Mechanisms

Some keystores offer other forms of substring or regular expression matching against the stored user IDs. These other forms of lookup may be useful in some contexts (e.g., Section 11.7), but they may also represent privacy concerns (e.g., Section 14.1), and they may impose additional computational or indexing burdens on the keystore.

3.4. Certificate Validation

An OpenPGP client may assess certificate and user ID validity based on many factors, some of which are directly contained in the certificate itself (e.g., third-party certifications), and some of which are based on the context known to the client, including:

- Whether it has seen e-mails from that address signed by that certificate in the past,

- How long it has known about the certificate,

- Whether the certificate was fetched from a keystore that asserts validity of the user ID or some part of it (such as the e-mail address).

A keystore MAY facilitate clients pursuing this last point of contextual corroboration via a direct interface:

```
keystore.cert_validate(primary_fpr, uid) -> boolean
```

In an e-mail-specific context, the client might only care about the keystore's opinion about the validity of the certificate for the e-mail address portion of the user ID only:

```
keystore.cert_validate(primary_fpr, addr) -> boolean
```

For some keystores, the presence of a certificate in the keystore alone implies that the keystore asserts the validity of all user IDs in the certificate retrieved. For others, the presence in the keystore applies only to some part of the user ID. For example, [PGP-GLOBAL-DIRECTORY] will only return user IDs that have completed an e-mail validation step, so presence in that keystore implies an assertion of validity of the e-mail address part of the user IDs returned, but makes no claim about the "display-name" portion of any returned user IDs. Note that a client retrieving a certificate from such a keystore may merge the certificate with a local copy - but the validity asserted by the keystore of course has no bearing on the packets that the keystore did not return.

In a more subtle example, the retrieval of a certificate looked up via WKD ([I-D.koch-openpgp-webkey-service]) or DANE ([RFC7929]) should only be interpreted as a claim of validity about any user ID which matches the e-mail address by which the certificate was looked up, with no claims made about any "display-name" portions, or about any user ID that doesn't match the queried e-mail address at all.

A keystore that offers some sort of validation interface may also change its opinion about the validity of a given certificate or user ID over time; the interface described above only allows the client to ask about the keystore's current opinion, but a more complex interface might be capable of describing the keystore's assertion over time. See also Section 11.5.

An abuse-resistant keystore that clients rely on for any part of their certificate validation process SHOULD offer a distinct interface for making assertions about certificate and user ID validity to help clients avoid some of the subtleties involved with inference based on presence described above.

Note that the certificate validation operation as described above has a boolean response. While a "true" response indicates that keystore believes the user ID or e-mail address is acceptable for use with the certificate referred to by the public key fingerprint, a "false" response doesn't necessarily mean that the keystore actively thinks that the certificate is actively bad, or must not be used for the referenced identity. Rather, "false" is the default state: no opinion is expressed by the keystore, and the client is left to make their own inference about validity based on other factors. A

keystore MAY offer a more nuanced validity interface; if it does, it SHOULD explicitly document the semantics of the different response types so that clients can make appropriate judgment.

3.5. Certificate Submission

Different keystores have different ways to submit a certificate for consideration for ingestion, including:

- a simple upload of a certificate via HTTP
- round-trip e-mail verification
- proof of presence in some other service
- vouching, or other forms of multi-party attestation

Because these schemes vary so widely, this document does not attempt to describe the keystore certificate submission process in detail. However, guidance can be found for implementations that generate, manage, and submit certificates in Section 10.

4. Simple Mitigations

These steps can be taken by any keystore that wants to avoid obviously malicious abuse. They can be implemented on receipt of any new packet, and are based strictly on the structure of the packet itself.

4.1. Decline Large Packets

While [RFC4880] permits OpenPGP packet sizes of arbitrary length, OpenPGP certificates rarely need to be so large. An abuse-resistant keystore SHOULD reject any OpenPGP packet larger than 8383 octets. (This cutoff is chosen because it guarantees that the packet size can be represented as a one- or two-octet [RFC4880] "New Format Packet Length", but it could be reduced further)

This may cause problems for user attribute packets that contain large images, but it's not clear that these images are concretely useful in any context. Some keystores MAY extend this limit for user attribute packets specifically, but SHOULD NOT allow even user attributes packets larger than 65536 octets.

4.2. Enforce Strict User IDs

[RFC4880] indicates that User IDs are expected to be UTF-8 strings. An abuse-resistant keystore MUST reject any user ID that is not valid UTF-8.

Some abuse-resistant keystores MAY only accept User IDs that meet even stricter conventions, such as an [RFC5322] "name-addr" or "addr-spec", or a URL like "ssh://host.example.org" (see Section 12.2).

As simple text strings, User IDs don't need to be nearly as long as any other packets. An abuse-resistant keystore SHOULD reject any user ID packet larger than 1024 octets.

4.3. Scoped User IDs

Some abuse-resistant keystores may restrict themselves to publishing only certificates with User IDs that match a specific pattern. For example, [RFC7929] encourages publication in the DNS of only certificates whose user IDs refer to e-mail addresses within the DNS zone. [I-D.koch-openpgp-webkey-service] similarly aims to restrict publication to certificates relevant to the specific e-mail domain.

4.4. Strip or Standardize Unhashed Subpackets

[RFC4880] signature packets contain an "unhashed" block of subpackets. These subpackets are not covered by any cryptographic signature, so they are ripe for abuse.

An abuse-resistant keystore SHOULD strip out all unhashed subpackets but the following exceptions:

4.4.1. Issuer Fingerprint

Some certifications only identify the issuer of the certification by an unhashed Issuer or Issuer Fingerprint subpacket. If a certification's hashed subpacket section has no Issuer Fingerprint (see [I-D.ietf-openpgp-rfc4880bis]) subpacket, then an abuse-resistant keystore that has cryptographically validated the certification SHOULD synthesize an appropriate Issuer Fingerprint subpacket and include it in the certification's unhashed subpackets.

4.4.2. Cross-sigs

Some Primary Key Binding Signatures ("cross-sigs") are distributed as unhashed subpackets in a Subkey Binding Signature. A cryptographically-validating abuse-resistant keystore SHOULD be willing to redistribute a valid cross-sig as an unhashed subpacket.

The redistributed unhashed cross-sig itself should be stripped of all unhashed subpackets.

4.4.3. First-party Attestations

Some third-party certifications are attested to by the certificate primary key itself in an unhashed subpacket, as described in Section 8.3. A cryptographically-validating abuse-resistant keystore SHOULD be willing to redistribute a valid first-party attestation as an unhashed subpacket.

The redistributed first-party attestation itself should be stripped of all unhashed subpackets.

4.5. Decline User Attributes

Due to size concerns, some abuse-resistant keystores MAY choose to ignore user attribute packets entirely, as well as any certifications that cover them.

4.6. Decline Non-exportable Certifications

An abuse-resistant keystore MUST NOT accept any certification that has the "Exportable Certification" subpacket present and set to 0. While most keystore clients will not upload these "local" certifications anyway, a reasonable public keystore that wants to minimize data has no business storing or distributing these certifications.

4.7. Decline Data From the Future

Many OpenPGP packets have time-of-creation timestamps in them. An abuse-resistant keystore with a functional real-time clock MAY decide to only accept packets whose time-of-creation is in the past.

Note that some OpenPGP implementations may pre-generate OpenPGP material intended for use only in some future window (e.g. "Here is the certificate we plan to use to sign our software next year; do not accept signatures from it until then."), and may use modified time-of-creation timestamps to try to achieve that purpose. This material would not be distributable ahead of time by an abuse-resistant keystore that adopts this mitigation.

4.8. Accept Only Profiled Certifications

An aggressively abuse-resistant keystore MAY decide to only accept certifications that meet a specific profile. For example, it MAY reject certifications with unknown subpacket types, unknown

notations, or certain combinations of subpackets. This can help to minimize the amount of room for garbage data uploads.

Any abuse-resistant keystore that adopts such a strict posture should clearly document what its expected certificate profile is, and should have a plan for how to extend the profile if new types of certification appear that it wants to be able to distribute.

Note that if the profile is ever restricted (rather than extended), and the restriction is applied to the material already present, such a keystore is no longer append-only (see Section 7).

4.9. Accept Only Certificates Issued by Designated Authorities

An abuse-resistant keystore capable of cryptographic validation MAY retain a list of designated authorities, typically in the form of a set of known public keys. Upon receipt of a new OpenPGP certificate, the keystore can decide whether to accept or decline each user ID of the certificate based whether that user ID has a certification that was issued by one or more of the designated authorities.

If no user IDs are certified by designated authority, such a keystore SHOULD decline the certificate and its primary key entirely. Such a keystore SHOULD decline to retain or propagate all certifications associated with each accepted user ID except for first-party certifications and certifications by the designated authorities.

The operator of such a keystore SHOULD have a clear policy about its set of designated authorities.

Given the ambiguities about expiration and revocation, such a keyserver SHOULD ignore expiration and revocation of authority certifications, and simply accept and retain as long as the cryptographic signature is valid.

Note that if any key is removed from the set of designated authorities, and that change is applied to the existing keystore, such a keystore may no longer be append-only (see Section 7).

4.10. Decline Packets by Blocklist

The maintainer of the keystore may keep a specific list of "known-bad" material, and decline to accept or redistribute items matching that blocklist. The material so identified could be anything, but most usefully, specific public keys or User IDs could be blocked.

Note that if a blocklist grows to include an element already present in the keystore, it will no longer be append-only (see Section 7).

Some keystores may choose to apply a blocklist only at retrieval time and not apply it at ingestion time. This allows the keystore to be append-only, and permits synchronization between keystores that don't share a blocklist, and somewhat reduces the attacker's incentive for flooding the keystore (see Section 5 for more discussion).

Note that development and maintenance of a blocklist is not without its own potentials for abuse. For one thing, the blocklist may itself grow without bound. Additionally, a blocklist may be socially or politically contentious as it may describe data that is toxic (Section 2.5) in one community or jurisdiction but not another. There needs to be a clear policy about how it is managed, whether by delegation to specific decision-makers, or explicit tests. Furthermore, the existence of even a well-intentioned blocklist may be an "attractive nuisance," drawing the interest of would-be censors or other attacker interested in controlling the ecosystem reliant on the keystore in question.

5. Retrieval-time Mitigations

Most of the abuse mitigations described in this document are described as being applied at certificate ingestion time. It's also possible to apply the same mitigations when a certificate is retrieved from the keystore (that is, during certificate lookup, refresh, or discovery). Applying an abuse mitigation at retrieval time may help a client defend against a user ID flooding (Section 2.2), certificate flooding (Section 2.1), or fingerprint flooding (Section 2.3) attack. It may also help a keystore limit its liability for redistributing toxic data (Section 2.5). However, only mitigations applied at ingestion time are able to mitigate keystore flooding attacks (Section 2.4).

Some mitigations (like the non-append-only mitigations described in Section 7) may be applied as filters at retrieval time, while still allowing access to the (potentially much larger) unfiltered dataset associated given certificate or user ID via a distinct interface.

The rest of this section documents specific mitigations that are only relevant at retrieval time (certificate discovery, lookup, or refresh).

5.1. Redacting User IDs

Some abuse-resistant keystores may accept and store user IDs but decline to redistribute some or all of them, while still distributing the certifications that cover those redacted user IDs. This draft refers to such a keystore as a "user ID redacting" keystore.

The certificates distributed by such a keystore are technically invalid [RFC4880] "transferable public keys", because they lack a user ID packet, and the distributed certifications cannot be cryptographically validated independently. However, an OpenPGP implementation that already knows the user IDs associated with a given primary key will be capable of associating each certification with the correct user ID by trial signature verification.

5.1.1. Certificate Refresh with Redacted User IDs

A user ID redacting keystore is useful for certificate refresh by a client that already knows the user ID it expects to see associated with the certificate. For example, a client that knows a given certificate currently has two specific user IDs could access the keystore to learn that one of the user IDs has been revoked, without any other client learning the user IDs directly from the keystore.

5.1.2. Certificate Discovery with Redacted User IDs

A user ID redacting keystore is somewhat less useful for clients doing certificate discovery. Consider the circumstance of receiving a signed e-mail without access to the signing certificate. If the verifier retrieves the certificate from a user ID redacting keystore by via the Issuer Fingerprint from the signature, and the signature validates, the received certificate might not be a valid "transferable public key" unless the client can synthesize the proper user ID.

A reasonable client that wants to validate a certification in the user ID redacted certificate SHOULD try to synthesize possible user IDs based on the value of the [RFC5322] From: header in the message:

- o Decode any [RFC2047] encodings present in the raw header value, converting into UTF-8 [UNICODE-NORMALIZATION] form C (NFC), trimming all whitespace from the beginning and the end of the string.

- o The resulting string should be an [RFC5322] "name-addr" or "addr-spec".

- o If it is a "name-addr", convert the UTF-8 string into an OpenPGP user ID and check whether the certification validates, terminating on success.

  - * If the test fails, extract the "addr-spec" from the "name-addr" and continue.

- Canonicalize the "addr-spec" according to Section 12.3, and check whether the certification validates, terminating on success.

- If it doesn't validate wrap the canonicalized "addr-spec" in angle-brackets ("<" and ">") and test the resulting minimalist "name-addr" against the certification, terminating on success.

- If all of the above fails, synthesis has failed.

5.1.3. Certificate Lookup with Redacted User IDs

It's possible (though non-intuitive) to use a user ID redacting keystore for certificate lookup. Since the keystore retains (but does not distribute) the user IDs, they can be used to select certificates in response to a search. The OpenPGP certificates sent back in response to the search will not contain the user IDs, but a client that knows the full user ID they are searching for will be able to verify the returned certifications.

Certificate lookup from a user ID redacting keystore works better for certificate lookup by exact user ID match than it does for substring match, because a client that retrieves a certificate via a substring match may not be able to reconstruct the redacted user ID.

However, without some additional restrictions on which certifications are redistributed (whether the user ID is redacted or not), certificate lookup can be flooded (see Section 13.1).

5.1.4. Hinting Redacted User IDs

To ensure that the distributed certificate is at least structurally a valid [RFC4880] transferable public key, a user ID redacting keystore MAY distribute an empty user ID (an OpenPGP packet of tag 13 whose contents are a zero-octet string) in place of the omitted user ID. This two-octet replacement user ID packet ("\xb4\x00") is called the "unstated user ID".

To facilitate clients that match certifications with specific user IDs, a user ID redacting keystore MAY insert a non-hashed notation subpacket into the certification. The notation will have a name of "uidhash", with 0x80 ("human-readable") flag unset. The value of such a notation MUST be 32 octets long, and contains the SHA-256 cryptographic digest of the UTF-8 string of the redacted user ID.

A certificate refresh client which receives such a certification after the "unstated user ID" SHOULD compute the SHA-256 digest of all user IDs it knows about on the certificate, and compare the result

with the contents of the "uidhash" notation to decide which user ID to try to validate the certification against.

5.1.5. User ID Recovery by Client Brute Force

User ID redaction is at best an imperfect process. Even if a keystore redacts a User ID, if it ships a certification over that user ID, an interested client can guess user IDs until it finds one that causes the signature to verify. This is even easier when the space of legitimate user IDs is relatively small, such as the set of commonly-used hostnames.

5.2. Primary-key Only Certificate Refresh

Abuse-resistant keystores can defend against a fingerprint flooding Section 2.3 attack during certificate refresh by implementing a narrowly-constrained certificate refresh interface.

Such a keystore MUST accept only a full fingerprint as the search parameter from the certificate refresh client, and it MUST return at most a single certificate whose primary key matches the requested fingerprint. It MUST NOT return more than one certificate, and it MUST NOT return any certificate whose primary key does not match the fingerprint. In particular, it MUST NOT return certificates where only the subkey fingerprint matches.

Note that [I-D.shaw-openpgp-hkp] does not offer the primitive described in Section 3.1 exactly. In that specification, the set of keys returned by a "get" operation with a "search" parameter that appears to be a full fingerprint is ambiguous. Some popular implementations (e.g., [SKS]) do not currently implement this mitigation, because they return certificates with subkeys that match the fingerprint.

5.3. Require Valid Cross-Sigs for Certificate Discovery

By definition, certificate discovery needs to be able to match subkeys, not just primary keys. This means that the mitigation in Section 5.2 is ineffective for a keystore that offers a certificate discovery interface.

An abuse-resistant keystore that aims to defend its certificate discovery interface from a fingerprint flooding (Section 2.3) attack can follow the following procedure.

Such a keystore MUST accept only a full fingerprint or a 64-bit key ID as the search parameter from the certificate discovery client. It MUST only match that fingerprint against the following:

- the fingerprint or key ID of a primary key associated with a valid certificate

- the fingerprint or key ID of a cryptographically-valid subkey that also has a cross-sig.

This defends against the fingerprint flooding attack because a certificate will only be returned by subkey if the subkey has agreed to be associated with the primary key (and vice versa).

Note that this mitigation means that certificate discovery will fail if used for subkeys that lack cross-sigs. In particular, this means that a client that tries to use the certificate discovery interface to retrieve a certificate based on its encryption-capable subkey (e.g., taking the key ID from a Public Key Encrypted Session Key (PKESK) packet) will have no success.

This is an acceptable loss, since the key ID in a PKESK is typically unverifiable anyway.

6. Contextual Mitigations

Some mitigations make the acceptance or rejection of packets contingent on data that is already in the keystore or the keystore's developing knowledge about the world. This means that, depending on the order that the keystore encounters the various material, or how it accesses or finds the material, the final set of material retained and distributed by the keystore might be different.

While this isn't necessarily bad, it may be a surprising property for some users of keystores.

6.1. Accept Only Cryptographically-verifiable Certifications

An abuse-resistant keystore that is capable of doing cryptographic validation MAY decide to reject certifications that it cannot cryptographically validate.

This may mean that the keystore rejects some packets while it is unaware of the public key of the issuer of the packet.

As long as the keystore implements the verification algorithm, Any self-signature should always be cryptographically-verifiable, since the public key of the issuer is already present in the certificate under consideration.

6.2. Accept Only Certificates Issued by Known Certificates

This is an extension of Section 4.9, but where the set of authorities is just the set of certificates already known to the keystore. An abuse-resistant keystore that adopts this strategy is effectively only crawling the reachable graph of OpenPGP certificates from some starting core.

A keystore adopting the mitigation SHOULD have a clear documentation of the core of initial certificates it starts with, as this is effectively a policy decision.

This mitigation measure may fail due to a compromise of any secret key that is associated with a primary key of a certificate already present in the keystore. Such a compromise permits an attacker to flood the rest of the network. In the event that such a compromised key is identified, it might be placed on a blocklist (see Section 4.10). In particular, if a public key is added to a blocklist for a keystore implementing this mitigation, and it is removed from the keystore, then all certificates that were only "reachable" from the blocklisted certificate should also be simultaneously removed.

FIXME: There are complexities associated with this strategy when certificates expire or are revoked. If expiration or revocation cause some certificates to become "unreachable", what should such a keystore do?

6.3. Rate-limit Submissions by IP Address

Some OpenPGP keystores accept material from the general public over the Internet. If an abuse-resistant keystore observes a flood of material submitted to the keystore from a given Internet address, it MAY choose to throttle submissions from that address. When receiving submissions over IPv6, such a keystore MAY choose to throttle entire nearby subnets, as a malicious IPv6 host is more likely to have multiple addresses.

This requires that the keystore maintain state about recent submissions over time and address. It may also be problematic for users who appear to share an IP address from the vantage of the keystore, including those behind a NAT, using a VPN, or accessing the keystore via Tor.

6.4. Accept Certificates Based on Exterior Process

Some public keystores resist abuse by explicitly filtering OpenPGP material based on a set of external processes. For example, [DEBIAN-KEYRING] adjudicates the contents of the "Debian keyring" keystore based on organizational procedure and manual inspection.

6.5. Accept Certificates by E-mail Validation

Some keystores resist abuse by declining any certificate until the user IDs have been verified by e-mail. When these "e-mail validating" keystores review a new certificate that has a user ID with an e-mail address in it, they send an e-mail to the associated address with a confirmation mechanism (e.g., a high-entropy HTTPS URL link) in it. The e-mail itself MAY be encrypted to an encryption-capable key found in the proposed certificate. If the keyholder triggers the confirmation mechanism, then the keystore accepts the certificate.

Some e-mail validating keystores MAY choose to distribute certifications over all user IDs for any given certificate, but will redact (see Section 5.1) those user IDs that have not been e-mail validated.

[PGP-GLOBAL-DIRECTORY] describes some concerns held by a keystore operator using this approach. [MAILVELOPE-KEYSERVER] is another example.

7. Non-append-only mitigations

The following mitigations may cause some previously-retained packets to be dropped after the keystore receives new information, or as time passes. This is entirely reasonable for some keystores, but it may be surprising for clients of a keystore that expect the keystore to be append-only (for example, some keyserver synchronization techniques may expect this property to hold).

Furthermore, keystores that drop old data (e.g., superseded certifications) may make it difficult or impossible for their users to reason about the validity of signatures that were made in the past. See Section 11.5 for more considerations.

Note also that many of these mitigations depend on cryptographic validation, so they're typically contextual as well.

A keystore that needs to be append-only, or which cannot perform cryptographic validation MAY omit these mitigations. Alternately, a keystore may omit these mitigations at certificate ingestion time,

but apply these mitigations at retrieval time (during certificate refresh, discovery, or lookup), and offer a more verbose (non-mitigated) interface for auditors, as described in Section 5.

Note that [GnuPG] anticipates some of these suggestions with its "clean" subcommand, which is documented as:

Compact (by removing all signatures except the selfsig) any user ID that is no longer usable (e.g. revoked, or expired). Then, remove any signatures that are not usable by the trust calculations. Specifically, this removes any signature that does not validate, any signature that is superseded by a later signature, revoked signatures, and signatures issued by keys that are not present on the keyring.

7.1. Drop Superseded Signatures

An abuse-resistant keystore SHOULD drop all signature packets that are explicitly superseded. For example, there's no reason to retain or distribute a self-sig by key K over User ID U from 2017 if the keystore have a cryptographically-valid self-sig over <K,U> from 2019.

Note that this covers both certifications and signatures over subkeys, as both of these kinds of signature packets may be superseded.

Getting this right requires a nuanced understanding of subtleties in [RFC4880] related to timing and revocation.

7.2. Drop Expired Signatures

If a signature packet is known to only be valid in the past, there is no reason to distribute it further. An abuse-resistant keystore with access to a functional real-time clock SHOULD drop all certifications and subkey signature packets with an expiration date in the past.

Note that this assumes that the keystore and its clients all have roughly-synchronized clocks. If that is not the case, then there will be many other problems!

7.3. Drop Dangling User IDs, User Attributes, and Subkeys

If enough signature packets are dropped, it's possible that some of the things that those signature packets cover are no longer valid.

An abuse-resistant keystore which has dropped all certifications that cover a User ID SHOULD also drop the User ID packet.

Note that a User ID that becomes invalid due to revocation MUST NOT be dropped, because the User ID's revocation signature itself remains valid, and needs to be distributed.

A primary key with no User IDs and no subkeys and no revocations MAY itself also be removed from distribution, though note that the removal of a primary key may make it impossible to cryptographically validate other certifications held by the keystore.

7.4. Drop All Other Elements of a Directly-Revoked Certificate

If the primary key of a certificate is revoked via a direct key signature, an abuse-resistant keystore SHOULD drop all the rest of the associated data (user IDs, user attributes, and subkeys, and all attendant certifications and subkey signatures). This defends against an adversary who compromises a primary key and tries to flood the certificate to hide the revocation.

Note that the direct key revocation signature MUST NOT be dropped.

In the event that an abuse-resistant keystore is flooded with direct key revocation signatures, it should retain the hardest, earliest revocation (see also Section 12.1).

In particular, if any of the direct key revocation signatures is a "hard" revocation, the abuse-resistant keystore SHOULD retain the earliest such revocation signature (by signature creation date).

Otherwise, the abuse-resistant keystore SHOULD retain the earliest "soft" direct key revocation signature it has seen.

If either of the above date comparisons results in a tie between two revocation signatures of the same "hardness", an abuse-resistant keystore SHOULD retain the signature that sorts earliest based on a binary string comparison of the direct key revocation signature packet itself.

7.5. Implicit Expiration Date

In combination with some of the dropping mitigations above, a particularly aggressive abuse-resistant keystore MAY choose an implicit expiration date for all signature packets. For example, a signature packet that claims no expiration could be treated by the keystore as expiring 3 years after issuance. This would permit the keystore to eject old packets on a rolling basis.

An abuse-resistant keystore that adopts this mitigation needs a policy for handling signature packets marked with an explicit expiration that is longer than implicit maximum. The two obvious strategies are:

- cap the packet's expiration to the system's implicit expiration date, or

- accept the explicit expiration date.

Warning: Any implementation of this idea is pretty radical, and it's not clear what it would do to an ecosystem that depends on such a keystore. It probably needs more thinking.

8. Primary Key Sovereignty

A keystore can defend against malicious external flooding by treating the "first party" of each certificate as "sovereign" over that certificate. This means in practice that no part of the certificate will redistributed without explicit permission from the primary key. We call a keystore that aims to respect primary key sovereignty a "sovereignty-respecting" keystore.

[RFC4880] defines "Key Server Preferences" with a "No-modify" bit. That bit has never been respected by any keyserver implementation that the author is aware of. A sovereignty-respecting keystore effectively treats that bit as always set, whether it is present in any part of the certificate or not.

A sovereignty-respecting abuse-resistant keystore can apply other constraints in addition to primary-key sovereignty, of course, for reasons as diverse as performance concerns, storage capacity, legal regulation, cryptographic algorithm support, or project policy. It will not redistribute anything that has not been explicitly approved by the primary key, but that does not mean it has to redistribute everything that has been explicitly approved by the primary key.

The remaining subsections of Section 8 describe some sensible strategies for a sovereignty-respecting keystore.

8.1. Refresh-only Keystores

Some soveriegnty-respecting keystores may resist abuse by declining to accept any user IDs or certifications whatsoever.

Such a keystore MUST be capable of cryptographic validation. It accepts primary key packets, cryptographically-valid direct-key signatures from a primary key over itself, subkeys and their

cryptographically-validated binding signatures (and cross-sigs, where necessary).

A client of a refresh-only keystore cannot possibly use the keystore for certificate lookup, because there are no user IDs to match. And it is not particularly useful for certificate discovery, because the returned certificate would have no identity information. However, such a keystore can be used for certificate refresh, as it's possible to ship revocations (which are direct key signatures), new subkeys, updates to subkey expiration, subkey revocation, and direct key signature-based certificate expiration updates.

Note that many popular OpenPGP implementations do not implement direct primary key expiration mechanisms, relying instead on user ID expirations. These user ID expiration dates or other metadata associated with a self-certification will not be distributed by an refresh-only keystore.

Certificates shipped by an refresh-only keystore are technically invalid [RFC4880] "transferable public keys," because they lack a user ID packet. However many OpenPGP implementations will accept such a certificate if they already know of a user ID for the certificate, because the composite certificate resulting from a merge will be a standards-compliant transferable public key.

8.2. First-party-only Keystores

Slightly more permissive than the refresh-only keystore described in Section 8.1 is a sovereignty-respecting keystore that also permits user IDs and their self-sigs.

A first-party-only keystore only accepts and distributes cryptographically-valid first-party certifications. Given a primary key that the keystore understands, it will only attach user IDs that have a valid self-sig, and will only accept and re-distribute subkeys that are also cryptographically valid (including requiring cross-sigs for signing-capable subkeys as recommended in [RFC4880]).

This effectively avoids certificate flooding attacks, because the only party who can make a certificate overly large is the holder of the secret corresponding to the primary key itself.

Note that a first-party-only keystore is still problematic for those people who rely on the keystore for retrieval of third-party certifications. Section 8.3 attempts to address this lack.

8.2.1. First-party-only Without User IDs

It is possible to operate an first-party-only keystore that redistributes certifications (in particular, self-sigs) while declining to redistribute user IDs themselves (see Section 5.1). This defends against concerns about publishing identifiable information, while enabling full certificate refresh for those keystore clients that already know the associated user IDs for a given certificate.

8.3. First-party-attested Third-party Certifications

We can augment a first-party-only sovereignty-respecting keystore to allow it to distribute third-party certifications as long as the first-party has signed off on the specific third-party certification.

A first-party can sign off a third-party certification by creating a signature over the third-party certification, and attaching it to the third-party certification as an unhashed subpacket. We call such a signature an "attestation", and it is created as detailed in the steps below:

- The cryptographic validity of the third-party certification SHOULD be verified by the first party before creating an attestation for it. This implies that the issuer key of the third-party certification must be known by the first party at the time when the attestation is created.

- The attestation MUST be an OpenPGP signature packet of type 0x50 (Third-Party Confirmation signature)

- The attestation MUST contain a hashed Signature Creation Time subpacket whose value is greater than or equal to the hashed Signature Creation Time subpacket of the third-party certification.

- The attestation MUST be a signature over the third-party certification, with all subpackets removed (see the discussion of signature type 0x50 in section 5.2.4 of [RFC4880]).

- The attestation MUST contain a hashed "Issuer Fingerprint" subpacket with the fingerprint of the primary key of the first-party certificate.

- The attestation MUST contain a hashed Notation subpacket with the name "first-party-attestation", and an empty (0-octet) value.

This attestation MUST be embedded in the third-party certification as an unhashed subpacket of type 32 (Embedded Signature). The simplest possible OpenPGP certificate with such an attestation looks like the following sequence of OpenPGP packets (indentation implies embedding):

- A. Primary key
- B. User ID
- C. Self-sig (from A, binding A to B)
- D. Third-party certification (from X, binding A to B)
  - E. (embedded unhashed in D: Attestation from A over D)

A sovereignty-respecting keystore MAY check the third-party certification itself for cryptographic validity, but explicitly MAY also omit this step as long as the first-party attestation is cryptographically valid. This allows distribution of any certificate as a self-contained unit by verifying only first-party signatures. It also allows distribution of third-party certifications made by third-parties unknown to the keystore.

A sovereignty-respecting keystore MUST only accept a third-party certification if it has a first-party attestation that is cryptographically valid, verifiable by the primary key of the certificate in question, and the first-party attestation meets all the constraints in the list above.

The Notation "first-party-attestation" is intended to avoid potential accidental confusion with any other use of the Third-Party Confirmation signature packet type. The author does not know of any current use that collides, but there may be future uses that we do not want to foreclose.

8.3.1. Client Interactions

Creating such an attestation requires multiple steps by different parties, each of which is blocked by all prior steps:

- The first-party creates the certificate, and transfers it to the third party.

- The third-party certifies it, and transfers their certification back to the first party.

- The first party attests to the third party's certification.

- Finally, the first party then transfers the compound certificate to the keystore.

The complexity and length of such a sequence may represent a usability obstacle to a user who needs a third-party-certified OpenPGP certificate.

No current OpenPGP client can easily create the attestations described in this section. More implementation work needs to be done to make it easy (and understandable) for a user to perform this kind of attestation.

8.3.2. Attestation Revocations

The certificate owner (the "first party") might at some point want to revoke their attestation over a given third-party certification. For example: Alice once was friends with Bob, but now thinks that Bob is a jerk, and wants to avoid public association with him. She used to be fine with keystores redistributing his certification of her cryptographic identity, but she no longer wants anyone to do that.

A sovereignty-respecting keystore that distributes third-party certifications SHOULD respect such a revocation by declining to distribute the associated third-party certification.

An "attestation revocation" is an OpenPGP signature packet of signature type 0x30 ("Certification revocation signature"), with the following properties:

- The attestation revocation MUST contain a hashed Signature creation Time subpacket whose value is greater than the attestation it revokes.

- The attestation revocation MUST contain a hashed "Issuer Fingerprint" subpacket with the fingerprint of the primary key of the first-party certificate.

- The attestation revocation MUST contain a hashed Notation subpacket with the name "first-party-attestation" and an empty (0-octet) value. This Notation MUST be marked as critical. The criticality is to avoid possible confusion with a revocation of the third-party certification itself, in the event that the third party has indicated that the first party is a designated revoker.

- Like any other revocation, the attestation revocation is made over the same data that the signature it is revoking was made over. In this case, that means: the third-party certification, with all subpackets removed (see the discussion of signature type 0x50 in section 5.2.4 of [RFC4880]).

- The attestation revocation SHOULD contain a hashed "Signature Target" subpacket with "public-key algorithm" that matches the public-key algorithm of the attestation. This subpacket makes it easier for clients and keystores alike to connect an attestation revocation with an already-known first-party attestation.

- The attestation revocation's hashed "Signature Target" subpacket's hash algorithm SHOULD be SHA256, and its value MUST be equal to the hash over the body of the attestation with all unhashed subpackets removed. That is, the material hashed should be the octet 0x88, followed by the four-octet length, followed by the body of the attestation with the unhashed subpacket data length set to zero.

- The attestation revocation SHOULD be shipped in the sequence of signature packets after the user ID (in the same location as the third-party certification itself). This is a different location than the attestation, for reasons described in Section 8.3.3.

If a keystore or a keystore client expects to see many first-party attested third-party certifications for a certificate, it SHOULD keep an index of them (the "attestation index"). To make effective use of the "Signature Target" subpacket, the attestation index should be keyed on the SHA256 hash of each indexed attestation, computed the same way it is computed for the "Signature Target" subpacket. Each entry in the attestation index should refer to both the attestation and its attested third-party certification. Note that SHA256 here is not used for its cryptographic strength, but merely as an efficiency measure for indexing.

8.3.3. Distribution of Attestation Revocations

The structure of an OpenPGP certificate with a revoked first-party attestation of a third-party certification looks like this series of OpenPGP packets (indentation implies embedding):

```
- A. Primary key
- B. User ID
- C. Self-sig (from A, binding A to B)
- D. Third-party certification (from X, binding A to B)
 - E. (embedded unhashed in D: Attestation from A over D)
- F. Attestation Revocation (from A over D, targets+revokes E)
```

A sovereignty-respecting keystore that sees all of these packets SHOULD respect Attestation Revocation F by not redistributing D and E at all.

However, if the keystore is distributing the certificate to a client that itself may care about first-party attestations (e.g. another sovereignty-respecting keystore, or just a sovereignty-respecting client), the keystore SHOULD continue to distribute F, so that the client can know that E has been revoked.

This means that the redistributed certificate will consist of this series of OpenPGP packets:

- A. Primary key
- B. User ID
- C. Self-sig (from A, binding A to B)
- F. Attestation Revocation (from A over D, target+revokes E)

A client or keystore that knows about D and E can verify the signature in F. It can identify E and D by looking them up in the attestation index based on the "Signature Target" subpacket in F. or, if it lacks such an index, it can scan all first-party attestations, and do trial signature verification over all third-party certifications it knows about until it finds a match.

A client or keystore that doesn't know about D or E MAY ignore and discard F, since it is not cryptographically verifiable without D and E. Maintaining an attestation index makes this kind of rejection much cheaper to compute.

A legacy client or keystore that doesn't understand attestations or attestation revocations at all might mistake F as an attempt at revocation of the User ID B itself, due to its signature type, its placement in the packet stream, and its issuer being A. However, such a client has three opportunities to ignore this packet and thereby avoid this confusion efficiently. In increasing order of computational effort:

o it has a critical notation set that the legacy client (by definition) does not understand, and should therefore reject.

o it has a Signature Target that does not match Self-sig C.

o it does not cryptographically validate when computed over A and B.

8.3.4. Revoking Third-party Certifications

A sovereignty-respecting keystore distributes a third-party certification based on the desires of the first party, but the third-party themselves may change their mind about the certification that they issued. In particular, they may revoke a previously attested third-party certification. This causes some additional complexity.

8.3.4.1. Third-party Certification Revocations Aren't Shipped with the Certificate

Distributing the third-party's revocation of their certification without the approval of the first party would arguably disrespect the first-party's sovereignty over their own certificate. For example, consider an abusively large revocation, or a revocation which contains toxic data.

At the same time, if the first party were to revoke their attestation, then the third-party certification itself _and_ its third-party's revocation might not be distributed. So distributing third-party certification revocations directly on the certificate they refer to doesn't seem to solve the problem for an abuse-resistant, sovereignty-respecting keystore.

8.3.4.2. Third-party Certification Revocations Ship With the Issuing Certificate

Instead, a sovereignty-respecting keystore MAY ship a third-party certification revocation attached to the end of the issuing certificate, as this respects the sovereignty of all parties involved.

Using the same packet references as Section 8.3, this means that the certifier's own OpenPGP certificate MAY be distributed like so:

- X. Primary key
- G. User ID
- H. Self-sig (from X, binding X to G)
- I. Subkey
- J. Subkey binding signature (from X, binds I to X)
- K. Certification revocation signature
  (from X over A and B, targets+revokes D)

Note that OpenPGP packet K is unusual here, and augments the traditional Transferable Public Key structure from [RFC4880].

A client that cares about third-party certifications SHOULD maintain an index of certifications based on the SHA256 digest of the certifications themselves (the "certification index"). The certification revocation packet SHOULD contain a Signature Target subpacket using SHA256 to identify the revoked certification. The client can use this Signature Target subpacket and the certification index to identify the targeted certification and to compute the data over which the revocation is made. As with the attestation index described in Section 8.3.2, this use of SHA256 is not used for cryptographic strength, but for indexing efficiency.

A client that cares about third-party certifications from key X SHOULD refresh the certificate containing X from the keystore, and verify any discovered certification revocations correctly to the appropriate certificates, searching for the targeted revocation in its certification index.

A legacy client that is unaware of this augmentation of the Transferable Public Key structure is likely to consider packet K as out-of-order or inapplicable (it would typically expect only a Subkey Revocation Signature packet in this position), and so will discard it.

In the event that the certificate has no subkeys (packets I and J are absent), a legacy client might consider K to be an attempt to revoke Self-Sig H. However, K's Signature Target subpacket does not point to H, and the certification is not cryptographically valid over X and G, so it should be discarded/ignored safely in that case as well.

9. Keystore Client Best Practices

An OpenPGP client that needs to interact with an abuse-resistant keystore can take steps to minimize the extent that its interactions with a keystore can be abused by other parties via the attacks described in Section 2. This section describes steps that an abuse-resistant client can take.

9.1. Use Constrained Keystores for Lookup

When performing certificate lookup, an abuse-resistant client SHOULD prefer to query abuse-resistant keystores to avoid the risks described in Section 13.1. In particular, keystores that defend against User ID Flooding are significantly more reliable for certificate lookup.

9.2. Normalize Addresses and User IDs for Lookup

When performing lookup by e-mail address, an abuse-resistant client SHOULD consider canonicalizing the e-mail address before searching (see Section 12.3).

When searching by full User ID, unless there is a strong reason to believe that a specific non-normalized form is preferable, an abuse-resistant client SHOULD normalize the entire user ID into [UNICODE-NORMALIZATION] Form C (NFC) before performing certificate lookup.

9.3. Avoid Fuzzy Lookups

Certificate lookup by arbitrary substring matching, or regular expression is prone to abuse. An abuse-resistant client SHOULD prefer exact-uid or exact-email match lookups where possible.

In particular, an abuse-resistant client should avoid trying to offer reliable functionality that performs these sort of fuzzy lookups, and SHOULD warn the user about risks of abuse if the user triggers a codepath that unavoidably performs such a fuzzy lookup.

9.4. Prefer Full Fingerprint for Discovery and Refresh

Key IDs are inherently weaker and easier to spoof or collide than full fingerprints. Where possible, an abuse-resistant keystore client SHOULD use the full fingerprint when interacting with the keystore.

9.5. Use Caution with Keystore-provided Validation

When an abuse-resistant client relies on a keystore for certificate validation, many things can go subtly wrong if the client fails to closely track the specific semantics of the keystore's validation claims.

For example, a certificate published by WKD ([I-D.koch-openpgp-webkey-service]) at "https://openpgpkey.example.org/.well-known/openpgpkey/hu/iy9q119eutrkn8s1mk4r39qejnbu3n5q?l=joe.doe" offers a validation claim only for the e-mail address "joe.doe@example.org". If the certificate retrieved at that address contains other user IDs, or if the user ID containing that e-mail address contains an [RFC5322] "display-name", none of that information should be considered "validated" by the fact that the certificate was retrieved via certificate lookup by WKD.

When certificate validation is represented more generally by a keystore via certificate retrieval (e.g. from an e-mail validating keyserver that has no distinct certificate validation interface), the thing validated is the certificate received from the keystore, and not the result of the merge into any local copy of the certificate already possessed by the client.

Consider also timing and duration of these assertions of validity, which represent a subtle tradeoff between privacy and risk as described in Section 14.4.

10. Certificate Generation and Management Best Practices

An OpenPGP implementation that generates or manages certificates and expects to distribute them via abuse-resistant keystores can take steps to ensure that the certificates generated are more likely to be accessible when needed. This section describes steps such an abuse-sensitive implementation can take.

10.1. Canonicalized E-Mail Addresses

E-mail addresses can be written in many different ways. An abuse-sensitive implementation considering attaching a user ID containing an e-mail address on a certificate SHOULD ensure that the e-mail address is structured as simply as possible. See Section 12.3 for details about e-mail address canonicalization.

For example, if the e-mail domain considers the local part to be case-insensitive (as most e-mail domains do today), if a proposed user ID contains the "addr-spec": "Alice@EXAMPLE.org", the implementation SHOULD warn the user and, if possible, propose replacing the "addr-spec" part of the user ID with "alice@example.org".

10.2. Normalized User IDs

User IDs are arbitrary UTF-8 strings, but UTF-8 offers several ways to represent the same string. An abuse-sensitive implementation considering attaching a user ID to a certificate SHOULD normalize the string using [UNICODE-NORMALIZATION] Form C (NFC) before creating the self-sig.

At the same time, the implementation MAY also warn the user if the "compatibility" normalized form (NFKC) differs from the candidate user ID and, if appropriate, offer to convert the user ID to compatibility normalized form at the user's discretion.

10.3. Avoid Large User Attributes

An abuse-sensitive implementation SHOULD warn the user when attaching a user attribute larger than 8383 octets, and SHOULD refuse to attach user attributes entirely larger than 65536 octets. (See Section 4.1)

10.4. Provide Cross-Sigs

[RFC4880] requires cross-sigs for all signing-capable subkeys, but is agnostic about the use of cross-sigs for subkeys of other capabilities.

An abuse-sensitive implementation that wants a certificate to be discoverable by subkey SHOULD provide cross-sigs for any subkey capable of making a cross-sig.

10.5. Provide Issuer Fingerprint Subpackets

Issuer subpackets contain only 64-bit key IDs. Issuer Fingerprint subpackets contain an unambiguous designator of the issuing key, avoiding the ambiguities described in Section 11.2. Abuse-sensitive implementations SHOULD provide Issuer Fingerprint subpackets.

10.6. Put Cross-Sigs and Issuer Fingerprint in Hashed Subpackets

Unhashed subpackets may be stripped or mangled. Placing cross-sigs and issuer fingerprint subpackets in the hashed subpackets will ensure that they are propagated by any cryptographically-validating keystore, even if that keystore fails to observe the exceptions in Section 4.4.

10.7. Submit Certificates to Restricted, Lookup-Capable Keystores

If an abuse-sensitive implementation wants other peers to be able to to retrieve the managed certificate by certificate lookup (that is, by searching based on user ID or e-mail address), it needs to be aware that submission to an unrestricted keystore is not reliable (see Section 13.1 for more details).

Consequently, such an implementation SHOULD submit the managed certificate to restricted, lookup-capable keystores where possible, as those keystores are more likely to be able to offer reliable lookup.

11. Side Effects and Ecosystem Impacts

11.1. Designated Revoker

A first-party-only keystore as described in Section 8.2 might decline to distribute revocations made by the designated revoker. This is a risk to certificate-holder who depend on this mechanism, because an important revocation might be missed by clients depending on the keystore.

FIXME: adjust this document to point out where revocations from a designated revoker SHOULD be propagated, maybe even in first-party-only keystores.

11.2. Key IDs vs. Fingerprints in Certificate Discovery

During signature verification, a user performing certificate discovery against a keystore SHOULD prefer to use the full fingerprint as an index, rather than the 64-bit key ID. Using a 64-bit key ID is more likely to run into collision attacks; and if the retrieved certificate has a matching key ID but the signature cannot be validated with it, the client is in an ambiguous state - did it retrieve the wrong certificate, or is the signature incorrect? Using the fingerprint resolves the ambiguity: the signature is incorrect, because the a fingerprint match is overwhelmingly stronger than a key ID match.

Unfortunately, many OpenPGP implementations distribute signatures with only an Issuer subpacket, so a client attempting to find such a certificate MAY perform certificate discovery based on only the key ID.

A keystore that offers certificate discovery MAY choose to require full fingerprint, but such a keystore will not be useful for a client attempting to verify a bare signature from an unknown party if that signature only has an Issuer subpacket (and no Issuer Fingerprint subpacket).

11.3. In-band Certificates

There are contexts where it is expected and acceptable that the signature appears without its certificate: for example, if the set of valid signers is already known (as in some OpenPGP-signed operating system updates), shipping the certificate alongside the signature would be pointless bloat.

However, OpenPGP signatures are often found in contexts where the certificate is not yet known by the verifier. For example, many OpenPGP-signed e-mails are not accompanied by the signing certificate.

In another example, the use of authentication-capable OpenPGP keys in standard SSH connections do not contain the full OpenPGP certificates, which means that the SSH clients and servers need to resort to out-of-band processes if evaluation of the OpenPGP certificates is necessary.

The certificate discovery interface offered by keystores is an attempt to accommodate these situations. But in the event that a keystore is unavailable, does not know the certificate, or suffers from a flooding attack, signature validation may fail unnecessarily.

In an encrypted e-mail context specifically, such a failure may also limit the client's ability to reply with an encrypted e-mail.

Certificate discovery also presents a potential privacy concern for the signature verifier, as noted in Section 14.5.

These problematic situations can be mitigated by shipping the certificate in-band, alongside the signature. Signers SHOULD adopt this practice where possible to reduce the dependence of the verifier on the keystores for certificate discovery. [AUTOCRYPT] is an example of e-mail recommendations that include in-band certificates.

11.3.1. In-band Certificate Minimization and Validity

OpenPGP certificates are potentially large. When distributing an in-band certificate alongside a signature in a context where size is a concern (e.g. bandwidth, latency, or storage costs are a factor), the distributor SHOULD reduce the size of the in-band certificate by stripping unnecessary packets. For example, the distributor may:

- Strip certification and signature packets that (due to creation and expiration time) are not relevant to the time of the signature itself. This ensures that the reduced certificate is contemporaneously valid with the signature.

- Strip irrelevant subkeys (and associated Subkey Binding Signature packets and cross-sigs). If the signature was issued by a signing-capable subkey, that subkey (and its binding signature and cross-sig) are clearly relevant. Other signing-capable subkeys are likely to be irrelevant. But determining which other subkeys are relevant may be context-specific. For example, in the e-mail context, an encryption-capable subkey is likely to be contextually relevant, because it enables the recipient to reply encrypted, and therefore should not be stripped.

- Strip user IDs (and associated certifications) that are unlikely to be relevant to the signature in question. For example, in the e-mail context, strip any user IDs that do not match the declared sender of the message.

- Strip third-party certifications that are unlikely to be relevant to the verifier. Doing this successfully requires some knowledge about what the third-parties the recipient is likely to care about. Stripping all third-party certifications is a simple means of certificate reduction. The verifier of such a certificate may need to do certificate refresh against their preferred keystore to learn about third-party certifications useful to them.

11.4. Certification-capable Subkeys

Much of this discussion assumes that primary keys are the only certification-capable keys in the OpenPGP ecosystem. Some proposals have been put forward that assume that subkeys can be marked as certification-capable. If subkeys are certification-capable, then much of the reasoning in this draft becomes much more complex, as subkeys themselves can be revoked by their primary key without invalidating the key material itself. That is, a subkey can be both valid (in one context) and invalid (in another context) at the same time. So questions about what data can be dropped (e.g. in Section 7) are much fuzzier, and the underlying assumptions may need to be reviewed.

If some OpenPGP implementations accept certification-capable subkeys, but an abuse-resistant keystore does not accept certifications from subkeys in general, then interactions between that keystore and those implementations may be surprising.

11.5. Assessing Certificates in the Past

Online protocols like TLS perform signature and certificate evaluation based entirely on the present time. If a certificate that signs a TLS handshake message is invalid now, it doesn't matter whether it was valid a week ago, because the present TLS session is the context of the evaluation.

But OpenPGP signatures are often evaluated at some temporal remove from when the signature was made. For example, software packages are signed at release time, but those signatures are validated at download time. A verifier that does not already know the certificate that made the signature will need to perform certificate discovery against some set of keystores to find a certificate with which to check the signature.

Further complicating matters, the composable nature of an OpenPGP certificate means that the certificate associated with any particular signing key (primary key or subkey) can transform over time. So when evaluating a signature that appears to have been made by a given certificate, it may be better to try to evaluate the certificate at the time the signature was made, rather than the present time.

11.5.1. Point-in-time Certificate Evaluation

When evaluating a certificate at a time T in the past (for example, when trying to validate a data signature by that certificate that was created at time T), one approach is to discard all packets from the certificate if the packet has a creation time later than T. Then

evaluate the resulting certificate from the remaining packets in the context of time T.

However, any such evaluation MUST NOT ignore "hard" OpenPGP key revocations, regardless of their creation date. (see Section 12.1).

11.5.2. Signature Verification and Non-append-only Keystores

If a non-append-only keystore (Section 7) has dropped superseded (Section 7.1) or expired (Section 7.2) certifications, it's possible for the certificate composed of the remaining packets to have no valid first-party certification at the time that a given signature was made. If a user performs certificate discovery against such a keystore, the certificate it retrieves would be invalid according to [RFC4880], and consequently verification of any signature by that certificate would fail.

One simple mitigation to this problem is to ship a contemporaneously-valid certificate in-band alongside the signature (see Section 11.3).

If the distributor does this, then the verifier need only learn about revocations. If knowledge about revocation is needed, the verifier might perform a certificate refresh (not "certificate discovery") against any preferred keystore, including non-append-only keystores, merging what it learns into the in-band contemporary certificate.

Then the signature verifier can follow the certificate evaluation process outlined in Section 11.5.1, using the merged certificate.

11.6. Global Append-only Ledgers ("Blockchain")

The append-only aspect of some OpenPGP keystores encourages a user of the keystore to rely on that keystore as a faithful reporter of history, and one that will not misrepresent or hide the history that they know about. An unfaithful "append-only" keystore could abuse the trust in a number of ways, including withholding revocation certificates, offering different sets of certificates to different clients doing certificate lookup, and so on.

However, the most widely used append-only OpenPGP keystore, the [SKS] keyserver pool, offers no cryptographically verifiable guarantees that it will actually remain append-only. Users of the pool have traditionally relied on its distributed nature, and the presumption that coordination across a wide range of administrators would make it difficult for the pool to reliably lie or omit data. However, the endpoint most commonly used by clients to access the network is "hkps://hkps.pool.sks-keyservers.net", the default for [GnuPG]. That endpoint is increasingly consolidated, and currently consists of

hosts operated by only two distinct administrators, increasing the risk of potential misuse.

Offering cryptographic assurances that a keystore could remain append-only is an appealing prospect to defend against these kinds of attack. Many popular schemes for providing such assurances are known as "blockchain" technologies, or global append-only ledgers.

With X.509 certificates, we have a semi-functional Certificate Transparency ([RFC6962], or "CT") ecosystem that is intended to document and preserve evidence of (mis)issuance by well-known certificate authorities (CAs), which implements a type of global append-only ledger. While the CT infrastructure remains vulnerable to certain combinations of colluding actors, it has helped to identify and sanction some failing CAs.

Like other global append-only ledgers, CT itself is primarily a detection mechanism, and has no enforcement regime. If a widely-used CA were identified by certificate transparency to be untrustworthy, the rest of the ecosystem still needs to figure out how to impose sanctions or apply a remedy, which may or may not be possible.

CT also has privacy implications - the certificates published in the CT logs are visible to everyone, for the lifetime of the log.

For spam abatement, CT logs decline all X.509 certificates except those issued by certain CAs (those in popular browser "root stores"). This is an example of the strategy outlined in Section 4.9).

Additional projects that provide some aspects of global append-only ledgers that try to address some of the concerns described here include [KEY-TRANSPARENCY] and [CONIKS], though they are not specific to OpenPGP. Both of these systems are dependent on servers operated by identity providers, however. And both offer the ability to detect a misbehaving identity provider, but no specific enforcement or recovery strategies against such an actor.

It's conceivable that a keystore could piggyback on the CT logs or other blockchain/ledger mechanisms like [BITCOIN] to store irrevocable pieces of data (such as revocation certificates). Further work is needed to describe how to do this in an effective and performant way.

11.7. Certificate Lookup for Identity Monitoring

While certificate lookup is classically used to find a key to encrypt an outbound message to, another use case for certificate lookup is

for the party in control of a particular identity to determine whether anyone else is claiming that identity.

That is, a client in control of the secret key material associated with a particular certificate with user ID X might search a keystore for all certificates matching X in order to find out whether any other certificates claim it.

This is an important safeguard as part of the ledger-based detection mechanisms described in Section 11.6, but may also be useful for keystores in general.

However, identity monitoring against a keystore that does not defend against user ID flooding (Section 2.2) is expensive and potentially of limited value. In particular, a malicious actor with a certificate which duplicates a given User ID could flood the keystore with similar certificates, hiding whichever one is in malicious use.

Since such a keystore is not considered authoritative by any reasonable client for the user ID in question, this attack forces the identity-monitoring defender to spend arbitrary resources fetching and evaluating each certificate in the flood, without knowing which certificate other clients might be evaluating.

12. OpenPGP details

This section collects details about common OpenPGP implementation behavior that are useful in evaluating and reasoning about OpenPGP certificates.

12.1. Revocations

It's useful to classify OpenPGP revocations of key material into two categories: "soft" and "hard".

If the "Reason for Revocation" of an OpenPGP key is either "Key is superseded" or "Key is retired and no longer used", it is a "soft" revocation.

An implementation that interprets a "soft" revocation will typically not invalidate signatures made by the associated key with a creation date that predates the date of the soft revocation. A "soft" revocation in some ways behaves like a non-overridable expiration date.

All other revocations of OpenPGP keys (with any other Reason for Revocation, or with no Reason for Revocation at all) should be considered "hard".

The presence of a "hard" revocation of an OpenPGP key indicates that the user should reject all signatures and certifications made by that key, regardless of the creation date of the signature.

Note that some OpenPGP implementations do not distinguish between these two categories.

A defensive OpenPGP implementation that does not distinguish between these two categories SHOULD treat all revocations as "hard".

An implementation aware of a "soft" revocation or of key or certificate expiry at time T SHOULD accept and process a "hard" revocation even if it appears to have been issued at a time later than T.

12.2. User ID Conventions

[RFC4880] requires a user ID to be a UTF-8 string, but does not constrain it beyond that. In practice, a handful of conventions predominate in how User IDs are formed.

The most widespread convention is a "name-addr" as defined in [RFC5322]. For example:

```
Alice Jones <alice@example.org>
```

But a growing number of OpenPGP certificates contain user IDs that are instead a raw [RFC5322] "addr-spec", omitting the "display-name" and the angle brackets entirely, like so:

```
alice@example.org
```

Some certificates have user IDs that are simply normal human names (perhaps "display-name" in [RFC5322] jargon, though not necessarily conforming to a specific ABNF). For example:

```
Alice Jones
```

Still other certificates identify a particular network service by scheme and hostname. For example, the administrator of an ssh host participating in the [MONKEYSPHERE] might choose a user ID for the OpenPGP representing the host like so:

```
ssh://foo.example.net
```

12.3. E-mail Address Canonicalization

When an OpenPGP user IDs includes an "addr-spec", there still may be multiple ways of representing the addr-spec that refer to the same underlying mailbox. Having a truly canonical form of an "addr-spec" is probably impossible because of legacy deployments of mailservers that do odd things with the local part, but e-mail addresses used in an abuse-resistant ecosystem SHOULD be constrained enough to admit to some sensible form of canonicalization.

12.3.1. Disallowing Non-UTF-8 Local Parts

In [RFC5322], the "local-part" can be any "dot-atom". But if this is [RFC2047] decoded, it could be any arbitrary charset, not necessarily UTF-8. FIXME: Do we convert from the arbitrary charset to UTF-8?

12.3.2. Domain Canonicalization

FIXME: should domain name be canonicalized into punycode form? User IDs are typically user-facing, so i think the canonicalized form should be the [UNICODE-NORMALIZATION] Form C (NFC) of the domain name. Can we punt to some other draft here?

12.3.3. Local Part Canonicalization

FIXME: [I-D.koch-openpgp-webkey-service] recommends downcasing all ASCII characters in the left-hand side, but leaves all

13. Security Considerations

This document offers guidance on mitigating a range of denial-of-service attacks on public keystores, so the entire document is in effect about security considerations.

Many of the mitigations described here defend individual OpenPGP certificates against flooding attacks (see Section 2.1). But only some of these mitigations defend against flooding attacks against the keystore itself (see Section 2.4), or against flooding attacks on the space of possible user IDs (see Section 2.2). Thoughtful threat modeling and monitoring of the keystore and its defenses are probably necessary to maintain the long-term health of the keystore.

Section 11.1 describes a potentially scary security problem for designated revokers.

TODO (more security considerations)

13.1. Tension Between Unrestricted Uploads and Certificate Lookup

Note that there is an inherent tension between accepting arbitrary certificate uploads and permitting effective certificate lookup. If a keystore accepts arbitrary certificate uploads for redistribution, it appears to be vulnerable to user ID flooding (Section 2.2), which makes it difficult or impossible to rely on for certificate lookup.

In the broader ecosystem, it may be necessary to use gated/controlled certificate lookup mechanisms. For example, both [I-D.koch-openpgp-webkey-service] and [RFC7929] enable the administrator of a DNS domain to distribute certificates associated with e-mail addresses within that domain, while excluding other parties. As a rather different example, [I-D.mccain-keylist] offers certificate lookup on the basis of interest - a client interested in an organization can use that mechanism to learn what certificates that organization thinks are worth knowing about, associated with a range of identities regardless of the particular DNS domain. Note that [I-D.mccain-keylist] does not provide the certificates directly, but instead expects the client to be able to retrieve them by primary key fingerprint through some other keystore capable of (and responsible for) certificate refresh.

14. Privacy Considerations

Keystores themselves raise a host of potential privacy concerns. Additional privacy concerns are raised by traffic to and from the keystores. This section tries to outline some of the risks to the privacy of people whose certificates are stored and redistributed in public keystores, as well as risks to the privacy of people who make use of the key stores for certificate lookup, certificate discovery, or certificate refresh.

14.1. Publishing Identity Information

Public OpenPGP keystores often distribute names or e-mail addresses of people. Some people do not want their names or e-mail addresses distributed in a public keystore, or may change their minds about it at some point. Append-only keystores are particularly problematic in that regard. The mitigation in Section 7.4 can help such users strip their details from keys that they control. However, if an OpenPGP certificate with their details is uploaded to a keystore, but is not under their control, it's unclear what mechanisms can be used to remove the certificate that couldn't also be exploited to take down an otherwise valid certificate.

Some jurisdictions may present additional legal risk for keystore operators that distribute names or e-mail addresses of non-consenting parties.

Refresh-only keystores (Section 8.1) and user ID redacting keystores (Section 5.1) may reduce this particular privacy concern because they distribute no user IDs at all.

14.2. Social Graph

Third-party certifications effectively map out some sort of social graph. A certification asserts a statement of belief by the issuer that the real-world party identified by the user ID is in control of the subject cryptographic key material. But those connections may be potentially sensitive, and some people may not want these maps built.

A first-party-only keyserver (Section 8.2) avoids this privacy concern because it distributes no third-party privacy concern.

First-party attested third-party certifications described in Section 8.3 are even more relevant edges in the social graph, because their bidirectional nature suggests that both parties are aware of each other, and see some value in mutual association.

14.3. Tracking Clients by Queries

Even without third-party certifications, the acts of certificate lookup, certificate discovery, and certificate refresh represent a potential privacy risk, because the keystore queried gets to learn which user IDs (in the case of lookup) or which certificates (in the case of refresh or discovery) the client is interested in. In the case of certificate refresh, if a client attempts to refresh all of its known certificates from the same keystore, that set is likely to be a unique set, and therefore identifies the client. A keystore that monitors the set of queries it receives might be able to profile or track those clients who use it repeatedly.

A privacy-aware client which wants to to avoid such a tracking attack MAY try to perform certificate refresh from multiple different keystores. To hide network location, a client making a network query to a keystore SHOULD use an anonymity network like [TOR]. Tools like [PARCIMONIE] are designed to facilitate this type of certificate refresh. Such a client SHOULD also decline to use protocol features that permit linkability across interactions with the same keystore, such as TLS session resumption, HTTP cookies, and so on.

Keystores which permit public access and want to protect the privacy of their clients SHOULD NOT reject access from clients using [TOR] or

comparable anonymity networks. Additionally, they SHOULD minimize access logs they retain.

Alternately, some keystores may distribute their entire contents to any interested client, in what can be seen as the most trivial form of private information retrieval. [DEBIAN-KEYRING] is one such example; its contents are distributed as an operating system package. Clients can interrogate their local copy of such a keystore without exposing their queries to a third-party.

14.4. "Live" Certificate Validation Leaks Client Activity

If a client relies on a keystore's certificate validation interface, or on the presence of a certificate in a keystore as a part of its validation calculations, it's unclear how long the assertion from the keystore is or should be considered to hold. One seemingly simple approach is to simply query the keystore's validation interface each time that the client needs to validate the certificate.

This "live" validation approach poses a quandary to the client in the event that the keystore is unavailable. How should in interpret the "unknown" result? In addition, live validation reveals the client's activity to the keystore with fine precision.

A privacy-aware client that depends on keystores for certificate validation SHOULD NOT perform "live" certificate validation on each use it makes of the certificate. Rather, it SHOULD cache the validation information for some period of time and make use of the cached copy where possible. If such a client does a regular certificate refresh from the same keystore, it SHOULD also pre-emptively query the keystore for certificate validation at the same time.

Choosing the appropriate time intervals for this kind of caching has implications for the windows of risk for the client that might use a revoked certificate. Defining an appropriate schedule to make this tradeoff is beyond the scope of this document.

14.5. Certificate Discovery Leaks Client Activity

The act of doing certificate discovery on unknown signatures offers a colluding keystore and remote peer a chance to track a client's consumption of a given signature.

An attacker with access to keystore logs could sign a message with a unique key, and then watch the keystore activity to determine when a client consumes the signature. This is potentially a tracking or "phone-home" situation.

A signer that has no interest in this particular form of tracking can mitigate this concern by shipping their certificate in-band, alongside the signature, as recommended in Section 11.3.

A privacy-aware client MAY insist on in-band certificates by declining to use any certificate discovery interface at all, and treat a bare signature by an unknown certificate as an invalid signature.

14.6. Certificate Refresh Leaks Client Activity

The act of doing certificate refresh itself reveals some information that the client is interested in a given certificate and how it may have changed since the last time the client refreshed it, or since it was first received by the client.

This is essentially the same privacy problem presented by OCSP [RFC6960] in the X.509 world. In the online world of TLS, this privacy leak is mitigated by the CertificateStatus TLS handshake extension ([RFC4366]), a.k.a. "OCSP stapling". There is no comparable solution for the store-and-forward or non-online scenarios where OpenPGP is often found.

Privacy-aware clients MAY prefer to access refresh interfaces from anonymity-preserving networks like [TOR] to obscure where they are on the network, but if the certificate being refreshed is known to be used only by a single client that may not help.

Privacy-aware clients MAY prefer to stage their certificate refreshes over time, but longer delays imply greater windows of vulnerability for use of an already-revoked certificate. This strategy also does not help when a previously-unknown certificate is encountered in-band (see Section 11.3), and the OpenPGP client wants to evaluate it for use in the immediate context.

14.7. Distinct Keystore Interfaces Leak Client Context and Intent

The distinct keystore interfaces documented in Section 3 offer subtly different semantics, and are used by a reasonable keystore client at different times. A keystore that offers distinct discovery and refresh interfaces may infer that a client visiting the refresh interface already knows about the certificate in question, or that a client visiting the discovery interface is in the process of verifying a signature from a certificate it has not seen before.

HKP itself ([I-D.shaw-openpgp-hkp]) conflates these two interfaces - the same HKP query is be used to perform both discovery and refresh (though implementations like [SKS] are not at all abuse-resistant for

either use), which may obscure the context and intent of the client from the keystore somewhat.

A privacy-aware client that can afford the additional bandwidth and complexity MAY use the keystore's discovery interface for both refresh and discovery, since the discovery interface is a proper superset of the refresh interface.

14.8. Cleartext Queries

If access to the keystore happens over observable channels (e.g., cleartext connections over the Internet), then a passive network monitor could perform the same type profiling or tracking attack against clients of the keystore described in Section 14.3. Keystores which offer network access SHOULD provide encrypted transport.

14.9. Traffic Analysis

Even if a keystore offers encrypted transport, the size of queries and responses may provide effective identification of the specific certificates fetched during lookup, discovery, or refresh, leaving open the types of tracking attacks described in Section 14.3. Clients of keystores SHOULD pad their queries to increase the size of the anonymity set. And keystores SHOULD pad their responses.

The appropriate size of padding to effectively anonymize traffic to and from keystores is likely to be mechanism- and cohort-specific. For example, padding for keystores accessed via the DNS ([RFC7929] may use different padding strategies that padding for keystores accessed over WKD ([I-D.koch-openpgp-webkey-service]), which may in turn be different from keystores accessed over HKPS ([I-D.shaw-openpgp-hkp]). A keystore which only accepts user IDs within a specific domain (e.g., Section 4.3) or which uses custom process (Section 6.4) for verification might have different padding criteria than a keystore that serves the general public.

Specific padding policies or mechanisms are out of scope for this document.

15. User Considerations

Section 8.3.1 describes some outstanding work that needs to be done to help users understand how to produce and distribute a third-party-certified OpenPGP certificate to an abuse-resistant keystore.

Additionally, some keystores present directly user-facing affordances. For example, [SKS] keyservers typically offer forms and listings that can be viewed directly in a web browser. Such a

keystore SHOULD be as clear as possible about what abuse mitigations it takes (or does not take), to avoid user confusion.

Keystores which do not expect to be used directly as part of a certificate validation calculation SHOULD advise clients as explicitly as possible that they offer no assertions of validity.

Experience with the [SKS] keyserver network shows that many users treat the keyserver web interfaces as authoritative. That is, users act as though the keyserver network offers some type of certificate validation. Unfortunately, The developer and implementor communities explicitly disavow any authoritative role in the ecosystem, and the implementations attempt very few mitigations against abuse, permitting redistribution of even cryptographically invalid OpenPGP packets. Clearer warnings to end users might reduce this kind of misperception. Or the community could encourage the removal of frequently misinterpreted user interfaces entirely.

16. IANA Considerations

This document asks IANA to register two entries in the OpenPGP Notation IETF namespace, both with a reference to this document:

- the "first-party-attestation" notation is defined in Section 8.3.

- the "uidhash" notation is defined in Section 5.1.4.

17. Document Considerations

[RFC Editor: please remove this section before publication]

This document is currently edited as markdown. Minor editorial changes can be suggested via merge requests at https://gitlab.com/dkg/draft-openpgp-abuse-resistant-keystore or by e-mail to the author. Please direct all significant commentary to the public IETF OpenPGP mailing list: openpgp@ietf.org

17.1. Document History

substantive changes bewteen -03 and -04:

- change "certificate update" to "certificate refresh" for clarity

- relax first-party-attested third-party certification constraints at the suggestion of Valodim

- introduce "primary key sovereignty" concept explicitly

- describe how to distribute and consume attestation revocations

- introduce augmentation to TPK for third-party certification revocation distribution

substantive changes between -02 and -03:

- new sections:

  * Keystore Interfaces

  * Keystore Client Best Practices

  * Certificate Generation and Management Best Practices

- rename "certificate discovery" to "certificate lookup"

- redefine "certificate discovery" to refer to lookup by signing (sub)key

- new attack: fingerprint flooding

- new retrieval-time mitigations - tighter filters on discovery and update

- recommend in-band certificates where possible to avoid discovery and lookup

- new privacy considerations:

  * distinct keystore interfaces

  * certificate update

  * certificate discovery

  * certificate validation

- more nuance about unhashed subpacket filtering

substantive changes between -01 and -02:

- distinguish different forms of flooding attack

- distinguish toxic data as distinct from flooding

- retrieval-time mitigations

- user ID redaction

- references to related work (CT, keylist, CONIKS, key transparency, ledgers/"blockchain", etc)

- more details about UI/UX

substantive changes between -00 and -01:

- split out Contextual and Non-Append-Only mitigations

- documented several other mitigations, including:

  - Decline Data From the Future

  - Blocklist

  - Exterior Process

  - Designated Authorities

  - Known Certificates

  - Rate-Limiting

  - Scoped User IDs

- documented Updates-Only Keystores

- consider three different kinds of flooding

- deeper discussion of privacy considerations

- better documentation of Reason for Revocation

- document user ID conventions

18. Acknowledgements

This document is the result of years of operational experience and observation, as well as conversations with many different people - users, implementors, keystore operators, etc. A non-exhaustive list of people who have contributed ideas or nuance to this document specifically includes:

- Antoine Beaupre

- ilf

- Jamie McClelland

- Jonathan McDowell

- Justus Winter

- Marcus Brinkmann

- Micah Lee

- Neal Walfield

- Phil Pennock

- Philihp Busby

- vedaal

- Vincent Breitmoser

- Wiktor Kwapisiewicz

19. References

19.1. Normative References

[I-D.ietf-openpgp-rfc4880bis]
Koch, W., carlson, b., Tse, R., and D. Atkins, "OpenPGP Message Format", draft-ietf-openpgp-rfc4880bis-07 (work in progress), May 2019.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, DOI 10.17487/RFC2047, November 1996, <https://www.rfc-editor.org/info/rfc2047>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

19.2. Informative References

[AUTOCRYPT] Breitmoser, V., Krekel, H., and D. Gillmor, "Autocrypt - Convenient End-to-End Encryption for E-Mail", n.d., <https://autocrypt.org/>.

[BITCOIN] "Bitcoin", n.d., <https://bitcoin.org/>.

[CONIKS] Felten, E., Freedman, M., Melara, M., Blankstein, A., and J. Bonneau, "CONIKS Key Management System", n.d., <https://coniks.cs.princeton.edu/>.

[DEBIAN-KEYRING] McDowell, J., "Debian Keyring", n.d., <https://keyring.debian.org/>.

[GnuPG] Koch, W., "Using the GNU Privacy Guard", April 2019, <https://www.gnupg.org/documentation/manuals/gnupg.pdf>.

[I-D.koch-openpgp-webkey-service] Koch, W., "OpenPGP Web Key Directory", draft-koch-openpgp-webkey-service-08 (work in progress), May 2019.

[I-D.mccain-keylist] McCain, R., Lee, M., and N. Welch, "Distributing OpenPGP Key Fingerprints with Signed Keylist Subscriptions", draft-mccain-keylist-04 (work in progress), March 2019.

[I-D.shaw-openpgp-hkp] Shaw, D., "The OpenPGP HTTP Keyserver Protocol (HKP)", draft-shaw-openpgp-hkp-00 (work in progress), March 2003.

[KEY-TRANSPARENCY] Belvin, G. and R. Hurst, "Key Transparency, a transparent and secure way to look up public keys", n.d., <https://keytransparency.org/>.

[MAILVELOPE-KEYSERVER] Oberndoerfer, T., "Mailvelope Keyserver", n.d., <https://github.com/mailvelope/keyserver/>.

[MONKEYSPHERE]
Gillmor, D. and J. Rollins, "Monkeysphere", n.d., <https://web.monkeysphere.info/>.

[PARCIMONIE]
Intrigeri, ., "Parcimonie", n.d., <https://gaffer.ptitcanardnoir.org/intrigeri/code/parcimonie/>.

[PGP-GLOBAL-DIRECTORY]
Symantec Corporation, "PGP Global Directory Key Verification Policy", 2011, <https://keyserver.pgp.com/vkd/VKDVerificationPGPCom.html>.

[RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, DOI 10.17487/RFC4366, April 2006, <https://www.rfc-editor.org/info/rfc4366>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/info/rfc6962>.

[RFC7929] Wouters, P., "DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP", RFC 7929, DOI 10.17487/RFC7929, August 2016, <https://www.rfc-editor.org/info/rfc7929>.

[SKS] Minsky, Y., Fiskerstrand, K., and P. Pennock, "SKS Keyserver Documentation", March 2018, <https://bitbucket.org/skskeyserver/sks-keyserver/wiki/Home>.

[TOR] "The Tor Project", n.d., <https://www.torproject.org/>.

[UNICODE-NORMALIZATION]
Whistler, K., "Unicode Normalization Forms", February 2019, <https://unicode.org/reports/tr15/>.

Author's Address

Daniel Kahn Gillmor
American Civil Liberties Union
125 Broad St.
New York, NY 10004
USA

Email: dkg@fifthhorseman.net